

T	P
6	6

Parte Teórica

1- El acoplamiento mide la dependencia de cada objeto en las definiciones de los demás. Un buen encapsulamiento reduce la interfaz accesible de cada objeto y por lo tanto limita las dependencias sobre este, dejando el posible acoplamiento.

2- Los modelos quedan poco ligados al mundo real, lo que suele producir una mala adaptabilidad a los cambios del problema y aumenta la carga cognitiva de su interpretación. Las clases impiden obtener más conocimiento del dominio, el nombre de clases.

3- Debemos intentar minimizar el acoplamiento y maximizar la cohesión, manteniendo la simplicidad del modelo en lo posible.

4- Evaluar un método implica que un objeto interprete el mensaje correspondiente, y siempre lo hará recordando su propio estado como contexto.

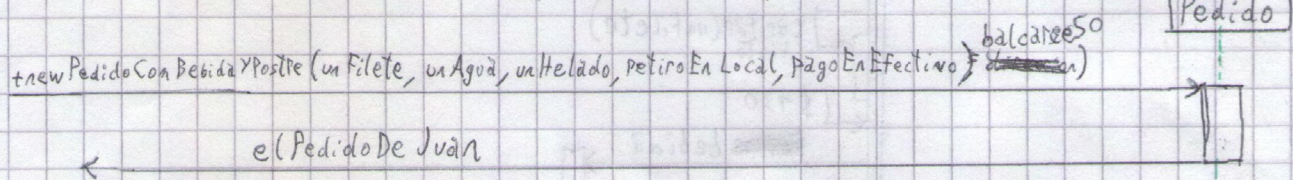
5- Se suelen usar closures para comportar funciones.
 R no son stacks

↓ Werners

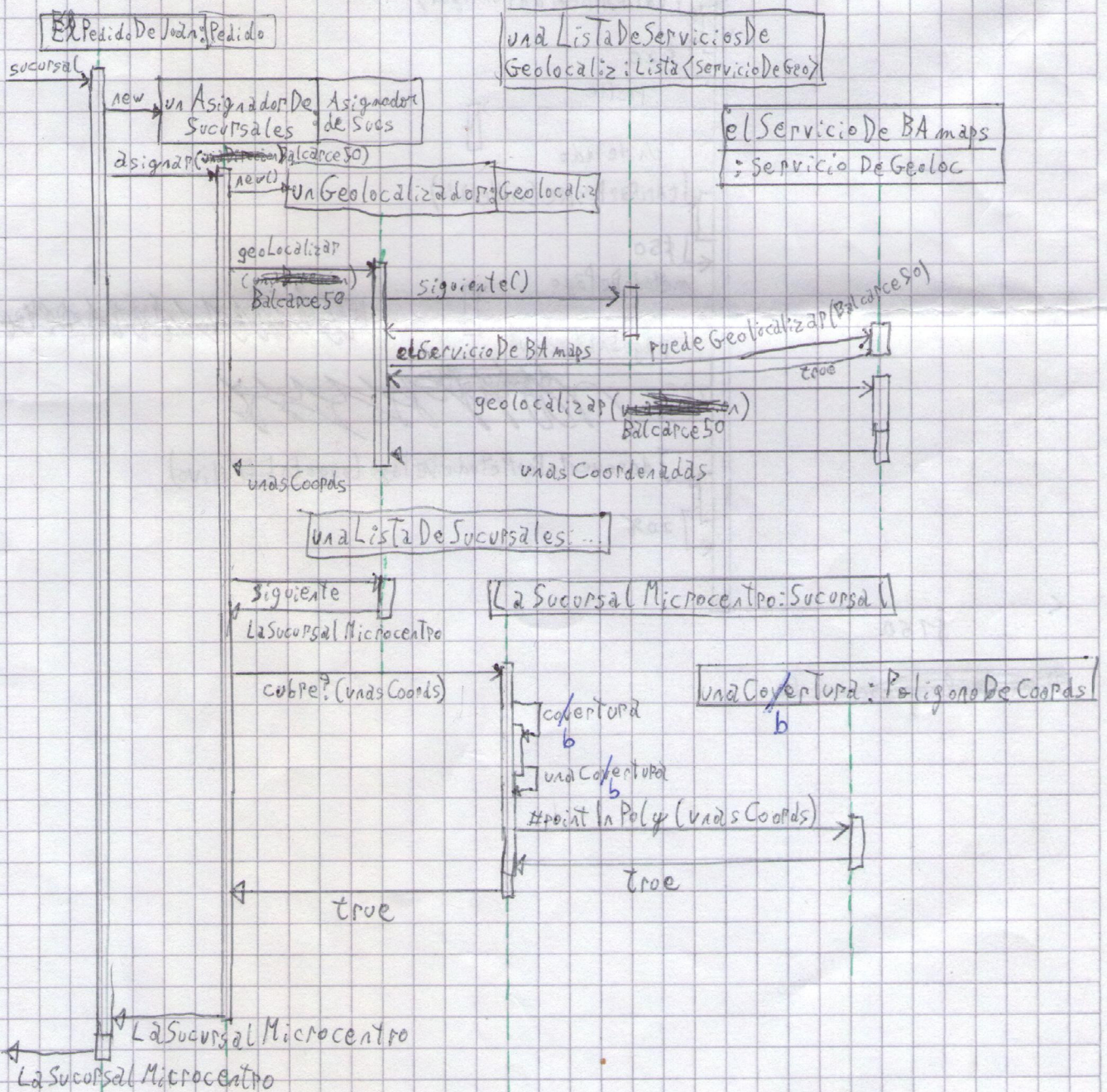
Parte Práctica

2. Diseño: Asumir que cada plato define si puede incluir postre

a) Cuando se crea el pedido se construye un Pedido que conoce los elecciones de Juan.



Cuando el sistema le muestra la sucursal asignada, le pregunta esta al pedido



Cuando se calcula el total a pagar, se le pregunta al Tarificador

un Tarificador De Pedidos: Tarif De Pedidos

total? (el Pedido De Juan)

el Pedido De Juan: Pedido

Plato

un Filete

tarifar(unFilete)

\$130

~~bebida~~

un Agua

tarifarBebida(unAgua)

\$20

Postre

un Helado

tarifarPostre(unHelado)

\$50

metodoDePago

pagoEnEfectivo

20%

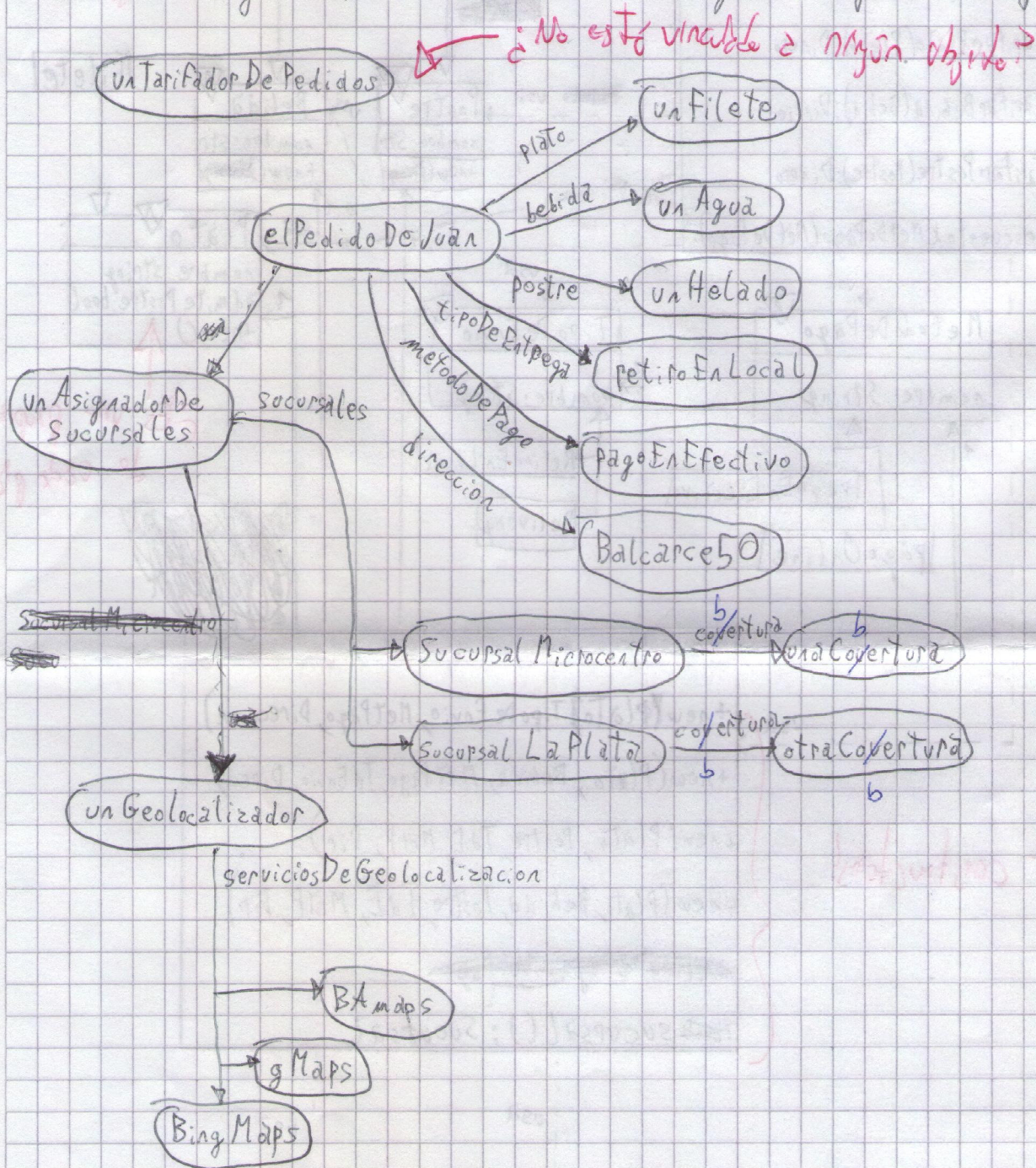
descuentoPorMetodoDePago(pagoEnEfectivo)

\$160

~~Durante la ejecución de~~

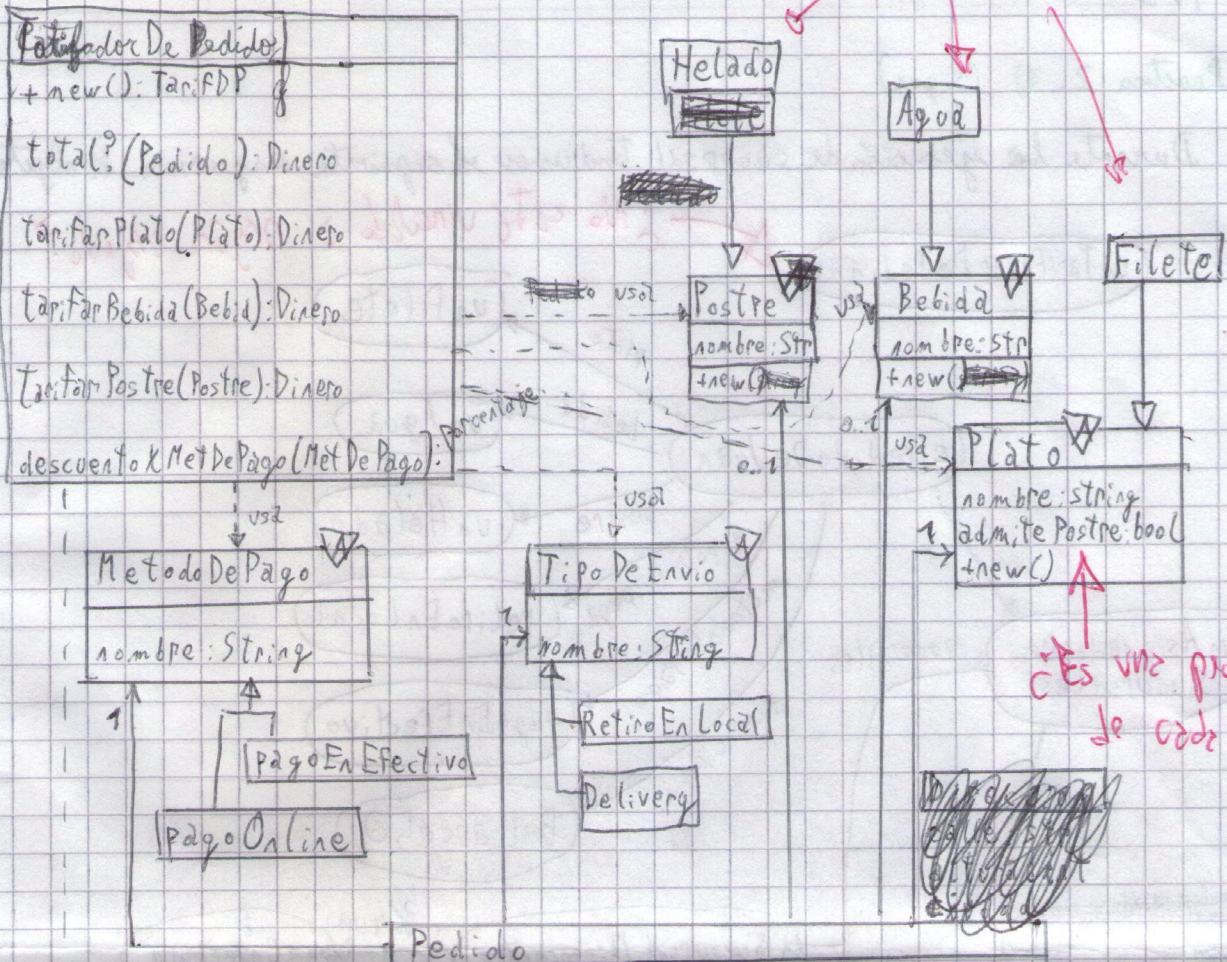
Práctica 2. a) (sigue)

Durante la ejecución de succursal tendremos el siguiente diagrama de objetos



DO no se componen de con DS: hay instrucciones entre Tarificador y pedido que no se reproducen en el DS.

b)



4 constructors!

```

+new(Plato, TipoDeEnvio, MetPago, Direccion)
+new(Plato, Bebida, MetPago, TdEnvio, Direc)
+new(Plato, Postre, TdE, MetP, Dir)
+new(Plato, Bebida, Postre, TdE, MetP, Dir)

```

~~metodoDePago~~

~~metodoDePago~~ sucursal(): Sucursal

AsignadorDeSucursales

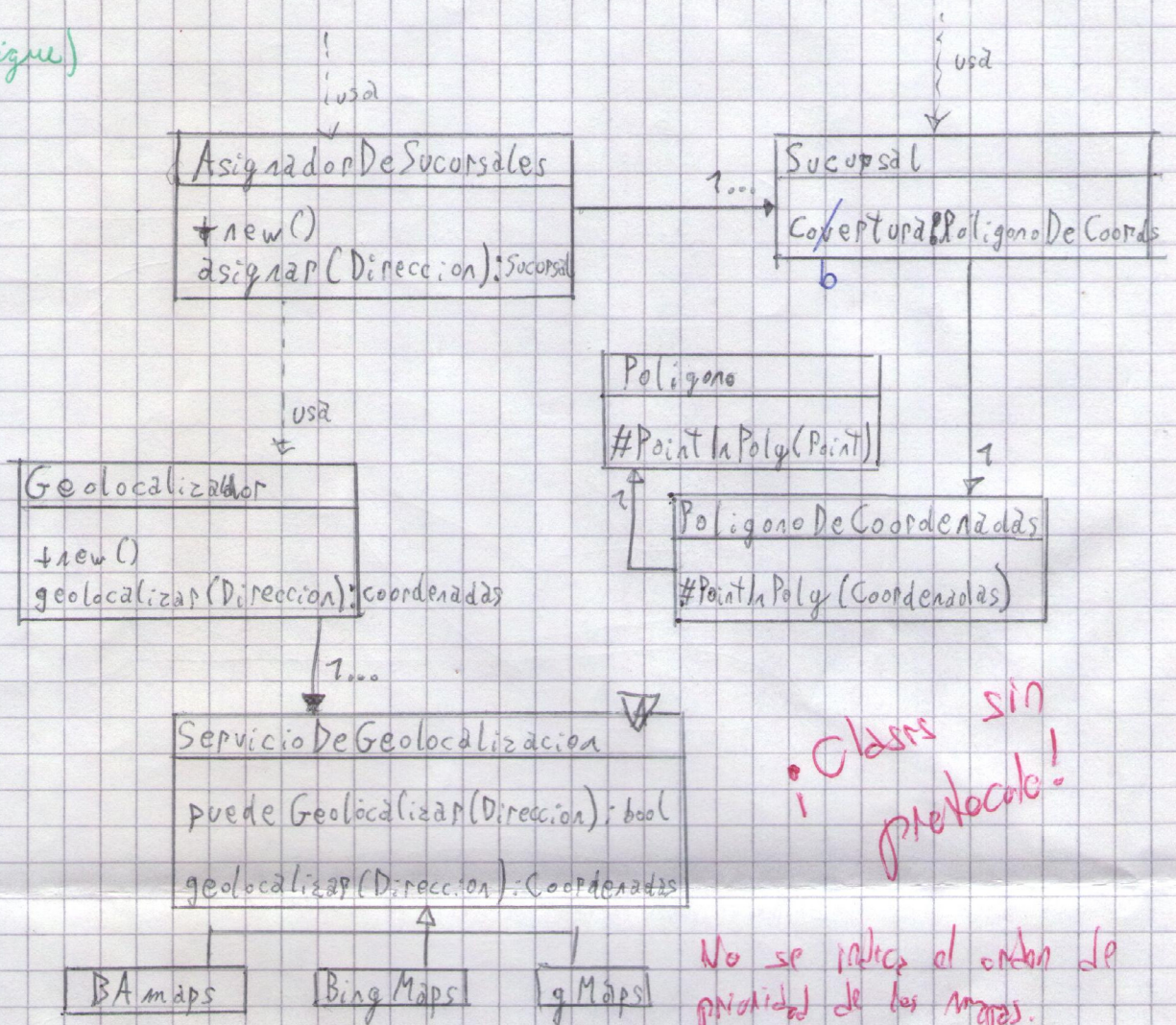
Sucursal

(definidos en la página siguiente)

OK pero ¿cómo se implementa?

* Pedido::Bebida y Pedido::Postre devolverán un Null Object si no corresponde.

b) (sigue)



X c) La clase **ServicioDeGeolocalizacion** funciona a modo de Bridge, generalizando la interfaz de ~~todos los~~ servicios para poder usarlos indistintamente. ! pero no aparece un Bridge en el diagrama de clases!

d) Con el diseño actual habría que usar un Tarifador De Pedidos diferente cada mes.

e) Habría que modificar Pedido para que ~~acepte~~ listas de platos, bebidas y postres y modificar el Tarifador para que los considere a todos.

f) Se podría agregar la opción de ~~ingresar directamente las coordenadas~~

Se podría generalizar la dirección para que se puedan indicar directamente las coordenadas, y el Asignador De Sucursales no tenga que geolocalizarlos.
 actualizar el teléfono